Performance Evaluation of an Object Management Policy Approach for P2P Networks

Dario Vieira, César Melo, Yacine Ghamri-Doudane

Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2012, Article ID 189325, 11 pages
doi:10.1155/2012/189325

Research Article

Performance Evaluation of an Object Management Policy Approach for P2P Networks

Dario Vieira,[1] Cesar A. V. Melo,[2] and Yacine Ghamri-Doudane[3,4]

[1] *LRIE Lab, École d'Ingénieur des Technologies de l'Information et de la Communication (EFREI), 94800 Villejuif, France*
[2] *Department of Computing Science, Federal University of Amazonas, 69077-000 Manaus, AM, Brazil*
[3] *École Nationale Supérieure d'Informatique pour l'Industrie et l'Entreprise (ENSIIE), 91025 Evry, France*
[4] *LIGM Lab, Université Paris-Est, 75420 Champs-sur-Marne, France*

Correspondence should be addressed to Dario Vieira, dario.vieira@efrei.fr

Received 1 June 2011; Revised 21 September 2011; Accepted 5 October 2011

Academic Editor: Ivan Lee

The increasing popularity of network-based multimedia applications poses many challenges for content providers to supply efficient and scalable services. Peer-to-peer (P2P) systems have been shown to be a promising approach to provide large-scale video services over the Internet since, by nature, these systems show high scalability and robustness. In this paper, we propose and analyze an object management policy approach for video web cache in a P2P context, taking advantage of object's metadata, for example, video popularity, and object's encoding techniques, for example, scalable video coding (SVC). We carry out trace-driven simulations so as to evaluate the performance of our approach and compare it against traditional object management policy approaches. In addition, we study as well the impact of churn on our approach and on other object management policies that implement different caching strategies. A YouTube video collection which records over 1.6 million video's log was used in our experimental studies. The experiment results have showed that our proposed approach can improve the performance of the cache substantially. Moreover, we have found that neither the simply enlargement of peers' storage capacity nor a zero replicating strategy is effective actions to improve performance of an object management policy.

1. Introduction

The increasing popularity of network-based multimedia applications poses many challenges for content providers to supply efficient and scalable services. Peer-to-peer (P2P) systems have been shown to be a promising approach to provide large-scale video services over the Internet since, by nature, these systems show high scalability and robustness.

One of the essential problems in measuring how these systems perform is the analysis of their properties in the presence of churn, a collective effect created by the independent arrival and departure of peers. Resiliency, key design parameters, and content availability are issues in P2P systems that are influenced by the churn. Hence, the user-driven dynamics of peer participation must be taken into account in both design and evaluation of any P2P system and its related mechanics, such as object management policies. For instance, when peers go off-line, the locally stored data become unavailable as well. This temporary content unavailability can lead to congestion on backbone links due to the extra traffic generated by requesting to the original content provider.

The main contributions of this paper are as follows.

(1) First, we propose and analyze an object management policy approach for videos' web cache in a P2P context. In this approach we exploit the object's metadata, for example, video popularity, and object's encoding techniques, for example, scalable video coding (SVC). In Section 5, we describe the object management policy based on popularity (POP); that is, we describe how user-generated content is used to define this object management policy. We have carried out set of studies by using three different scenarios so as to analyze our approach with regarding other object management policies. We evaluated how the insertion, replacement, and discarding of videos impacts

the object management policies. Besides, we study as well the effects of the video popularity and the discarding of video layers in the performance of object management policies. In Section 6 we present the numerical results collected from these simulation experiments. We show as well how much our content-oriented web cache mechanism can improve traditional web cache mechanisms. Basically, we have found that our approach outperforms the traditional one and consequently reduce the capacity demand poses over the community output link by increasing the hit rate of community demands and optimizing the network resources available.

(2) Second, we study the impact of the user churn on object management policies. For that, we have used a churn model that is based on a stationary alternating renewal process with its average up and down behavior based on the type of peer (cf. Section 4). By using this churn model, we have evaluated the enlargement of individual peer storage capacity as an action to keep policies performance since the community storage capacity will be affected directly by the churn. In Section 7, we present the numerical results collected from these simulation experiments. We can point that peers will be able to store valuable objects over a long time-scale which could improve content availability. We have noted that the gap among all police performance shrinks due to this enlargement. Nonetheless, this enlargement does not impact linearly the performance of the policies.

(3) Finally, we have evaluated on how much the replicated data affects policies performance by measuring it on a system with and without duplicated data. Indeed, each time a peer leaves the system, it will make chunks of popular content unavailable for other peers. Whenever that peer rejoins the system, its content could have been accessed by other peer. Accordingly, data are naturally replicated into the system due to the churn. This naturally replication has as consequence the decreases of the nominal system storage capacity. So, we have evaluated on how much this replicated data affects policies performance by measuring it on a system with and without duplicated data. In Section 7.1, we present the numerical results collected from these simulation experiments. We have found that the performance of policies is impacted, either positively or negatively, by this replicated data which suggests that content availability could be improved whether the volume of duplicated data is under policy control.

We have used in our experimental studies a YouTube video collection, which records over 1.6 million video's log. In our experiments, each peer can store up to 25 short videos of 4:50 minutes—the average video length identified in the studied video collection. The overall storage capacity, which is defined by the sum of the storage capacity of all peers, is equal to one percent of the video collection storage demand (cf. Section 3). Different simulated scenarios were defined by scaling up the system storage capacity until 20% of the video collection storage demand.

We have evaluated five policies in our experiments: (i) the context-aware and content-oriented policy (POP), (ii) least recently used (LRU) policy, (iii) least frequently used (LFU) policy, (iv) popularity-aware greedy-dual size (GDSP) policy [1], and (v) proportional partial caching (PARTIAL) policy [2]. The last four policies (described in Section 2) are representative implementations of their classes, that is, recency-based, frequency-based, and cost-based policies.

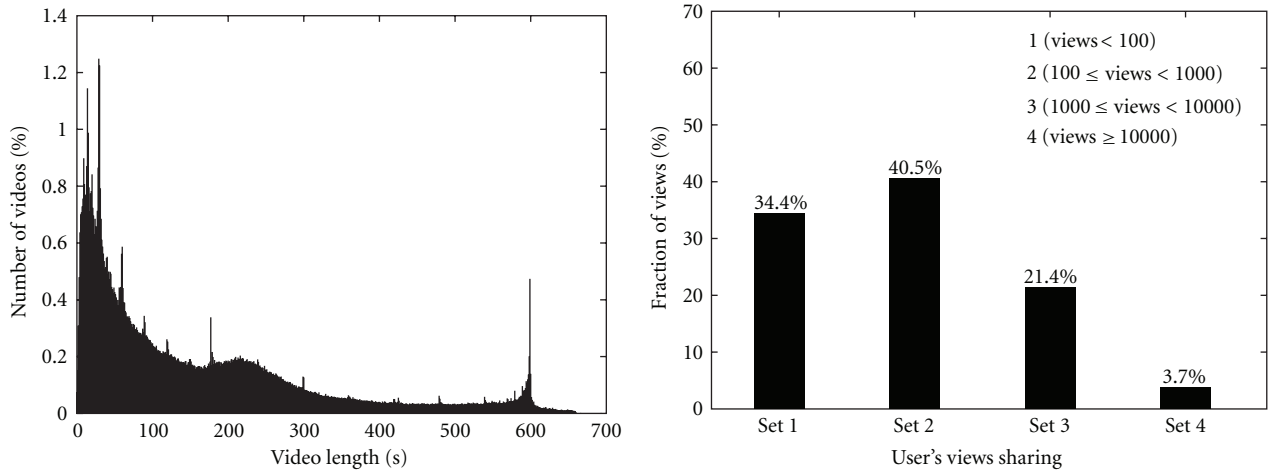
2. Cache Algorithms

This section gives a short overview of the traditional cache algorithms that we use through this paper. Essentially, these algorithms can be classified into three major flavors of strategies as follows.

Recency-Based Strategies. This kind of approach uses recency as a main factor. Least recently used (LRU) strategy, and all its extensions, is a good example of these strategies. LRU makes use of the locality information to predict future accesses to objects from past accesses. In effect, there are two sorts of locality: (i) temporal locality and (ii) spatial locality. The first one is based on the idea that recently accessed objects are likely to be used again in the future. In the latter approach, the references to some objects suggest accesses to other objects. Recency-based strategies make use of temporal locality; that is, LRU exploits temporal locality. The major disadvantage of these strategies is that they do not consider object size information. Moreover, they do not consider frequency information.

Frequency-Based Strategies. These strategies exploit the frequency as a main factor. They are based on the idea that different objects have different popularity values and this implies that these objects have different frequency values. Accordingly, these values are used for future decisions. The least-frequently used (LFU) is the well-known implementation of these strategies. There are two kinds of implementation of LFU (and its extensions): (i) perfect LFU, which counts all requests to an object. This keeps on across replacement; (ii) in-cache LFU, whose counts are carried out only for cache objects. It should be noted that this does not represent all request in the past. The major disadvantage is that the frequency counts are too static for dynamic environments.

Cost-Based Strategies. In cost-based policies, the importance of an object is defined by a value got from a cost function. When a cache runs out of space, objects are evicted based on their cost; objects with the smallest cost will be evicted first. The greedy-dual size (GDS) policy was the first policy to implement a cost-based cache strategy. It maintains, for each object, a characteristic value Hi which is set in the very first request and calculated every time the object is requested. Improvements on GDS policy have been implemented to



(a) Videos length (b) Popularity

Figure 1: Statistics for YouTube Entertainment category [3].

consider historical and access frequency data (GDSP—popularity-aware greedy-dual size [1]) and the dynamics of peer-to-peer system (proportional partial caching) [2].

3. Video Collection

In this section, we present the video collection used so as to carry out our experiments. Our dataset consists of meta-information about user-generated videos from YouTube services. We limited our experiments to the *Entertainment* category collected by [3] owing to the massive scale of YouTube. This collection consists of 1,687,506 videos where each line represents a single video. Furthermore, each video record contains both fixed and time-varying information (e.g., length, views, ratings, stars, and URL), which means

(1) views and ratings stand for the number of times the video has been played and evaluated by users,

(2) stars stand for the average score from rating, and

(3) URL denotes the list of external web pages hyperlinking the video.

We limited the maximum size of cacheable video to 99 minutes; videos with more than this value are considered crawler mistake. Figure 1(a) depicts videos' length distribution in seconds, with video length being equal to 291 seconds.

Figure 1(b) shows the video popularity distribution. By examining the number of requests recorded in our video collection, we grouped those videos in four sets. In the first set we gathered video with less than 100 views. This subset is made of 34.4% of videos in our collection. In the second subset we gathered videos with a number of views into the range of 100 views and 1000 views, which contains 40.5% of videos in our collection. The third subset has videos with a number of views into the range of 1,000 views and 10,000 views and makes up 21.4% of videos recorded in our collection. Finally, in the fourth subset we gathered videos that recorded over 10.000 views, which contains 3.7% of videos in our collection. Based on our analysis, we see that video requests are highly concentrated on a small set of videos. In fact, the fourth subset has 60% of the total number of views.

This video collection misses individual user requests information; that is, timestamps mark that record when a user dispatched his/her requests. To deploy our simulation studies, we must have such individual user's behavior. Hence, we developed a procedure to simulate the users' behavior based on information gathered from the video collection. Therefore, in order to use this collection, we define a procedure for the generation of requests, as follows.

(1) All requests will be driven by a combination of the video length and the "think time."

(2) For each two nonpopular videos, we picked up three popular videos. This ratio was derived based on the preview analysis carried over our video collection; that is, we determined the number of (non)popular video views recorded.

(3) The pick-up procedure of popular and not popular videos is independent and follows a uniform distribution.

(4) A popular video must be recorded more than 10,000 views.

The rationales behind our procedure are (i) downloaded videos will be watched end-to-end, and users will spend some time looking for related videos, the thinking time, before dispatching a new request; (ii) over the time scale the system is observed, popular and nonpopular videos will keep their status, a reasonable assumption since we are interested only on the effectiveness of popularity as a criterion to manage those videos; (iii) in our collection, videos with more than 10,000 views represent less than 4% of the whole collection but recorded over 60% of the total number of views.

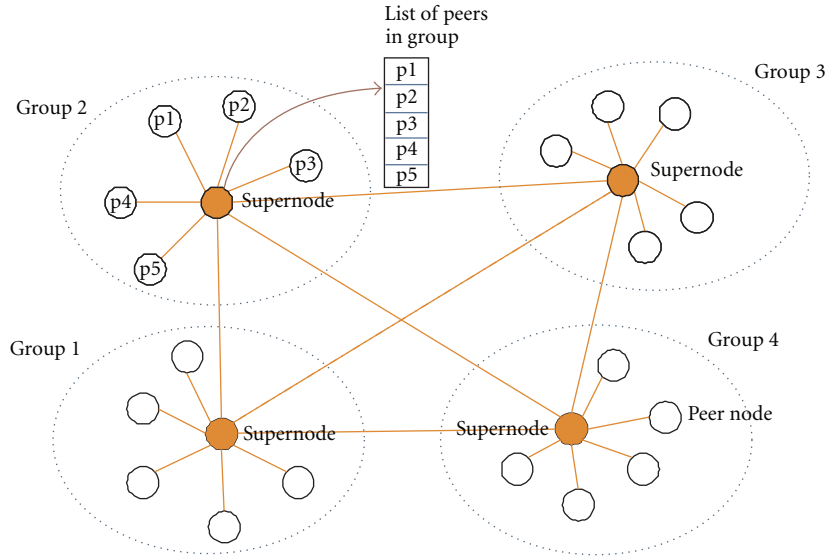


Figure 2: Two-tier hierarchy framework.

4. A Hierarchical Content-Oriented Overlay Network

In this section, we present the networking scenario used in our studies. First, the network infrastructure is considered, then we describe how peers are classified and objects are searched in this infrastructure. Finally, the churn model used to characterize peers' dynamics is presented.

4.1. The Network Infrastructure. The network infrastructure is a structured peer-to-peer (P2P) system made by ordinary nodes and supernodes and with peers that are locality-aware clustered into communities or groups. These communities are built around subjects, for example, science, sports, and DIY. Figure 2 illustrates this infrastructure. Peers locality awareness limits content lookups to very few hops into the community. In Crespo and Garcia-Molina [4], similar ideas have been proposed to share music files with overlays being built based on music genres, and lookup operations being bounded to the community.

In this network infrastructure, for each group there is one or more superpeers, which are analogous to gateways routers in hierarchical IP networks. The goal is to use superpeers so as to transmit messages intergroups.

On this networking scenario, the mix of gathered peers defines a community as homogeneous or heterogeneous. Accordingly, peers can be classified based on their up probability and they can be clustered in $\mathcal{T} > 1$ groups. In this paper, we classify and cluster them in four groups (i.e., $\mathcal{T} = 4$) as follows.

(i) *Stable group*, each peer is up with probability big or equal to p.

(ii) *Moderate group*, each peer is up with probability t, but with $p \gg t$.

(iii) *Low group*, each peer is up with probability r, but with $t \gg r$.

(iv) *Unstable group*, each peer is up with probability less or equal to q, but with $r \gg q$.

These peer group's up probabilities are mapped to the peer group's up session duration in our numerical studies as suggested by Wang et al. [5]. Specifically, we studied a community made by peers that fits in a low group behavior; that is, peers up sessions last 28 minutes in average, and a typical YouTube session duration by the time our video collection was collected, according to Wang et al. [5].

4.2. The Churn Model. To model the peers' dynamics, we have exploited a generic churn model defined by Yao et al. [6]. Consider a P2P system with n peers, where each peer i is either UP at time t or DOWN. This behavior can be modeled by a renewal process $\{Z_i(t)\}$ for each peer i:

$$Z_i(t) = \begin{cases} 1, & \text{peer } i \text{ is alive at time } t, \\ 0, & \text{otherwise,} \end{cases} \quad 1 \le i \le n. \tag{1}$$

Unlike [6], the UP and DOWN lasting sessions of $\{Z_i(t)\}$ are based on the type of peer i. Therefore, the actual pairs $(F_i(x), G_i(x))$ are chosen randomly from set $\mathcal{F}$ define as follows.

$$\mathcal{F} = \left\{ \left(F^{(1)}(x), G^{(1)}(x)\right), \ldots, \left(F^{(\mathcal{T})}(x), G^{(\mathcal{T})}(x)\right) \right\}, \tag{2}$$

where $\mathcal{T} \ge 1$ is the number of peer types in the system.

Therefore, for each process $\{Z_i(t)\}$, its UP lasting sessions $\{L_{i,c}\}_{c=1}^{\infty}$ have some joint distribution $F_i(x)$, and its DOWN lasting sessions $\{D_{i,c}\}_{c=1}^{\infty}$ have another joint distribution $G_i(x)$, where c stands for cycle number and durations of users and i's UP and DOWN sessions are given by random variables $L_{i,c} > 0$ and $D_{i,c} > 0$, respectively.

Examples of UP/DOWN distributions used throughout this paper are (i) the exponential, defined as

$$F_i' = 1 - e^{-\lambda_i x}, \tag{3}$$

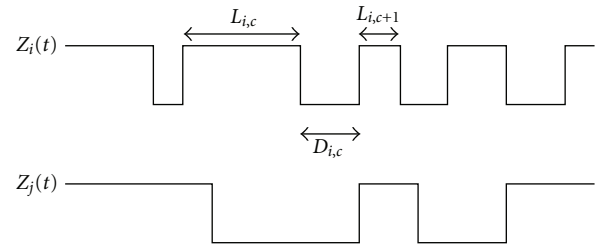
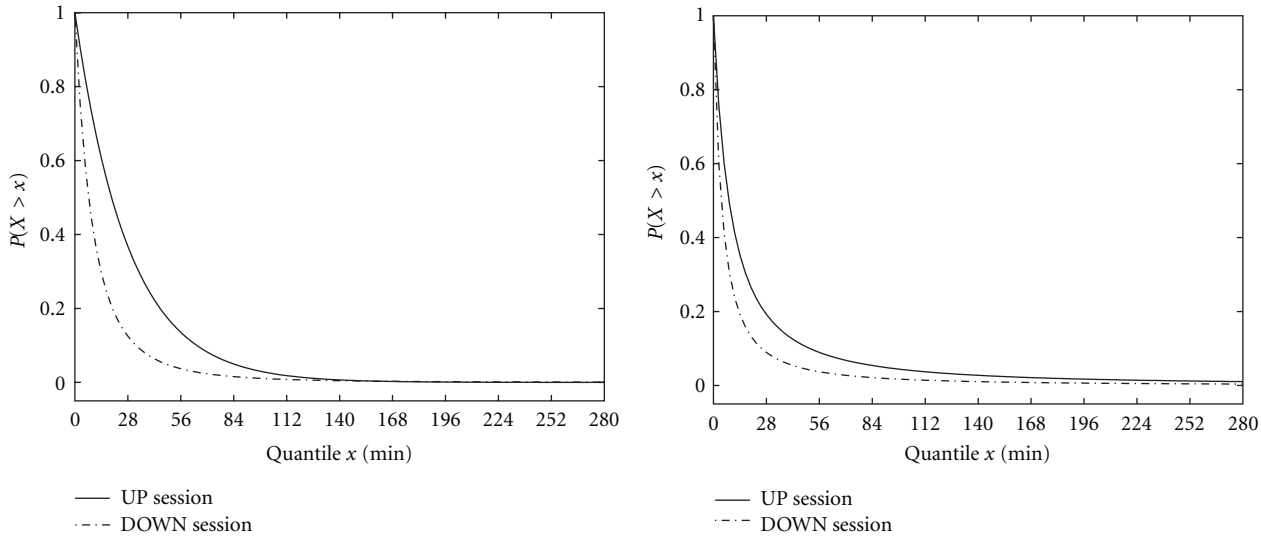


FIGURE 3: Up and down behaviors for peer p_i and p_j. Processes $\{Z_i(t)\}$ and $\{Z_j(t)\}$ are independent.



(a) Exponential churn model

(b) Heavy tail churn model

FIGURE 4: Complementary cumulative distribution functions of UP and DOWN session duration used on exponential and heavy tail churn models.

with mean $1/\lambda_i$, and (ii) the Pareto, defined as

$$F_i^{''} = 1 - \left(1 + \frac{x}{\beta_i}\right)^{\alpha i}, \quad x > 0,\ \alpha_i > 1, \tag{4}$$

with mean $\beta_i/(\alpha_i - 1)$.

Based on the aforementioned model, the following assumptions are made.

(1) To capture the independent nature of peers, we assume that the set $\{Z_i(t)\}_{i=1}^{n}$ consists of mutually independent alternating renewal processes, as depicted in the Figure 3. Accordingly, peers behave independently of each other and processes $\{Z_i(t)\}$ and $\{Z_j(t)\}$, for any $i \neq j$, are independent.

(2) Note that l_i, the average UP session duration, and d_i, the average DOWN session duration, are independent and unique for each peer. Once pair $(l_i; d_i)$ is generated for each peer p_i, it remains constant for the entire evolution of the system.

(3) 28-minute YouTube average session duration is set to l_i. In addition, half of the value of l_i is set to d_i. Hence, as regards content availability, a demanding community of peers is expected.

(4) The upprobability of peer i at an arbitrary time instance t is given by

$$p = \lim_{t \to \infty} P(\{Z_i(t)\} = 1) = \frac{l_i}{l_i + d_i}. \tag{5}$$

Based on previous characterization, two churn models have been used on our studies: heavy tail and exponential. When exponential churn model is applied, the lasting of UP session is driven by $\exp(1/l_i)$ distribution, while the lasting of DOWN session is driven by a *pareto* $(3, d_i)$ distribution, where l_i stands for the expected mean of UP session duration and d_i stands for the expected mean DOWN session duration.

Figure 4(a) shows the complementary cumulative distribution function (CCDF) of the exponential (UP session) and Pareto (DOWN session) distributions when l_i and d_i are set as mentioned previously. For exponential distribution, the probability of chose values greater than the expected ones vanishes after minute 120. However, for Pareto distribution, this probability still exists over large time-scale. This churn pattern can be summarized by the following: peers will stay connected, as they join the community, but will lose interest over time.

```
Require: video_popularity ∨ cache_size ≥ 0
    request_from_community(video);
    if miss_video then
        request_from_outside(video);
    end if
    if video_popularity > threshold then
        insert_cache(video);
    end if
```

Algorithm 1: The popularity-based object management policy.

In the heavy tail churn model, the UP and DOWN sessions are driven by the Pareto distributions. Different from exponential churn model, heavy tail churn model draws probability still significant over time, Figure 4(b), which means that picking a value great than the mean value is highly possible. The churn pattern defined by this heavy tail model can be summarized by the following: peers will keep interest on content over long time-scales.

5. POP: A Content-Oriented Management Policy Based on Popularity

As mentioned in Section 4.1, peers will cooperate and the interesting-oriented community has very useful information to improve the cache system performance. In this context, the probability that a video comes to be accessed again by other members of a community is higher than that in a general case since members inside a community might share common interests. Hence, the question about how much popular a video is inside a community seems to be an important metadata to be considered when implementing an object management policy.

Based on the aforementioned assumptions, we have developed an object management policy [7], that is, the popularity-based (POP) policy. In this object management policy, we keep the most popular video in cache (number of visualizations) based on a predefined threshold. The rationale of our policy is that the majority of video requests is targeted to the popular ones, hence keeping the cache filled by the popular video will probably improve the hit rate and decrease the volume of traffic that flows outside the community link. Algorithm 1 describes this procedure.

As said, the threshold used to identify popular videos is a predefined value and closed related to video collection statistics. In our studies we set the threshold equal to 10,000 visualizations which defines a popular video collection with less than 4% of the whole video collection, at the same time, this popular video collection has over 60% of all visualizations. In summary, the threshold setting procedure will reduce the number of videos that have to be managers, but those videos will receive the majority of requests.

6. Performance Analysis of POP

In this section we show the numerical results collected from our first studies. We evaluated how the insertion, replacement, and discarding of videos impact the object management policies. To evaluate the effectiveness of our proposed policy, we defined a reference scenario and we compared it to three other different scenarios so as to evaluate the performance of the POP. For that, we assume that peers are always connected to the system, a borrowed concept from [5] which identifies and describes the importance of stable peers.

```
Require: video_popularity ∨ cache_size ≥ 0
    if video_popularity > threshold then
        insert_cache(video);
    end if
```

Algorithm 2: Considering video popularity.

6.1. Numerical Results. We use a trace-driven simulation to evaluate our proposed policy. The simulated P2P network has 10,000 peers, and we assume that there is only one community where each peer can connect to any other peer. Each peer has a cache with capacity to storage up to 1.000 seconds of video. The total number of video requests is 200,000, and we consider that videos with more than a *threshold value* are popular. In our experiments, the threshold value is equal to 10,000 views.

6.2. Reference Scenario. In this scenario, we establish the following conditions: (i) every requested (and not yet cached) video must be added to the cache, (ii) LRU is the object replacement policy implemented by the web cache, and (iii) the whole video will be discarded when the managed cache is full. Taking into consideration these assumptions, we simulated the user requests based on a uniform distribution. As we have pointed out, the results are then compared with three other scenarios, which are described as follows.

6.3. Scenario Number 1. In this scenario, we evaluated the effects of the video popularity on the performance of our web cache. Instead of adding all referenced, and not yet cached videos, we keep only cached videos that are considered popular, that is, videos that have a number of views greater than 10,000. The rationale of our policy is that the majority of video requests is targeted to the popular ones, hence keeping the cache filled by the popular video will probably improve the hit rate and decrease the volume of traffic that flow outside the community link. In summary, in scenario number 1 we cached only popular videos, the object management policy is still the LRU, and the whole video is discarded when the managed cache is full.

Algorithm 2 describes the patched applied to the object management policy; that is, the execution of function *insert_cache(·)* is conditioned to the video popularity. Figures 5 and 6, the first bars, show the improvements, in terms of hit rate (35.2%) and volume of traffic that is saved (36.2%), when that new policy is implemented.

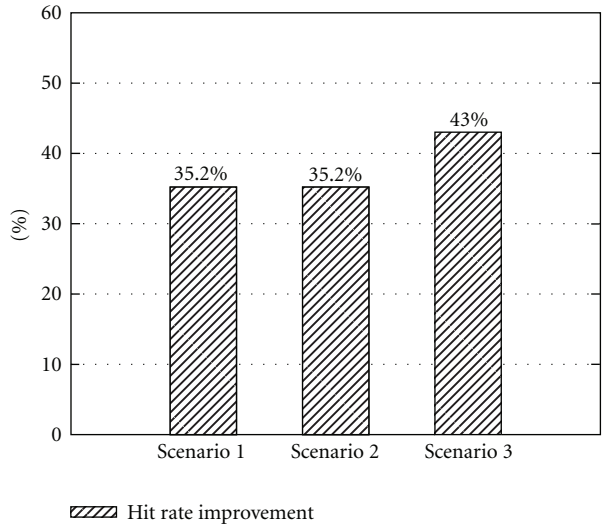


Figure 5: Hit rate—based on reference scenario.

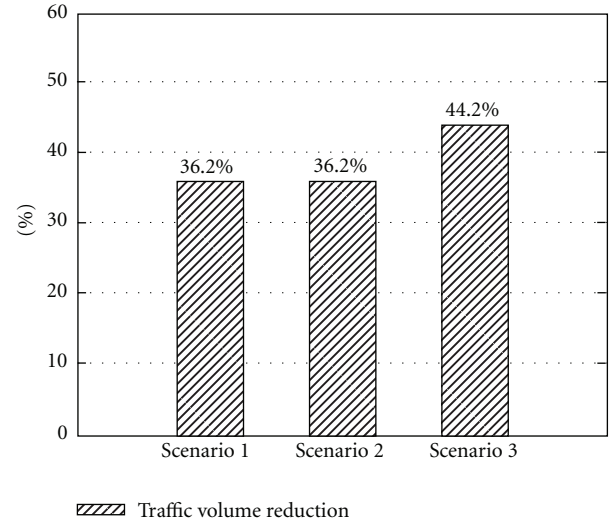


Figure 6: Traffic volume reduction—based on reference scenario.

```
Require: free_cache_size ∨ video_size
  While free_cache_size < video_size do
    (video,node) ⇐ LRU_video_object();
    nlayers ⇐ discard_highest_layer(video, node);
    if nlayers = 0 then
      nlayers ⇐ discard_highest_layer(video, node);
      remove_reference_DHT(video, node);
    end if
  end while
```

Algorithm 3: Discarding layers, the local approach.

```
Require: free_cache_size ∨ video_size
  While free_cache_size < video_size do
    (video, node) ⇐ random_video_object();
    nlayers ⇐ discard_highest_layer(video, node);
    if nlayers = 0 then
      nlayers ⇐ discard_highest_layer(video, node);
      remove_reference_DHT(video, node);
    end if
  end while
```

Algorithm 4: Discarding layers, the global approach.

6.4. Scenario Number 2. In this scenario, we evaluated the effects of the video popularity and the discarding of video layers on the performance of our object management policy. Differently to the scenario number 1, we dispose only video layers that are coding to improve the video quality and hence keep those layers that have the minimum amount of data to the receiver reproduce his/her requested video. The least recently used criterion is applied to elect which video must have its layers discarded. Whether the room made available by the discarded layer is still insufficient to store the recently accessed video, a new video is elected and its layer is discarded. This process is repeated until the web cache has sufficient space to store the recently referenced video. The rationale of our policy is to keep as much as possible a popular video cached, even it is a low-quality copy, since references to it are very probable.

Algorithm 3 presents the patch applied to object management policy. Function *LRU_video_object(·)* finds the least recently used cached object and returns both that video and the node that has the object. Function *discard_highest_layer (video, node)* discards the highest layer of the found video. For example, if there are two layers for this video in cache, the second, which has less priority, will be discarded. Additionally, after calling the *discard_highest_layer(·)* function no layers could remain for that video, hence the reference for this video must be removed from distributed hash table (DHT). Figures 5 and 6, the second bars, show the improvements in term of hit rate (35.2%) and saved traffic (36.2%) when a new policy is implemented.

6.5. Scenario Number 3. In this scenario, we introduce the layer discarding policy associated with a global approach instead of the local one deployed in scenario number 2. In other words, we randomly selected objects in cache and discarded the highest layer of this object. The rationale of our approach is that when a cache run out of space, we have a set of high popular videos cached with all those video showing a high requesting rate. Hence when we apply the discarding policy over the whole set of videos, and not just over the "least" recently used set, we will increase the set of popular videos that can be cached. Consequently, we can observe an improvement over the hit rate. In summary, scenario number 3 has only popular video cached, a random object replacement policy, and video layers are discarded when cache is full.

Algorithm 4 presents the patches applied to the object management policy. Function *random_video_object(·)* picks a cached object and returns both the targeted video and node that has that object.

Figures 5 and 6, the third bars, show the improvements in terms of hit rate (43%) and saved traffic (44.2%) when the new policy is implemented.

6.6. Discussions. Requests to popular video are much more probe than nonpopular videos; over 60% of requests recorded in our video collection are targeted to popular videos. In addition, popular videos made just 3.7% of our video collection. Based on these facts, we tested the video popularity metadata as the criteria to cache or dispose recently accessed videos. As noted from scenario number 1, there was a 30% improvement in the cache hit rate, compared to the traditional approach, reassuring that bringing metadata associated to the objects will save the network resources with server communities that were built around subjects.

Although 30% improvement is remarkable, we learnt that traditional web cache policies do not assume anything about the managed objects. In the scenario number 3, we exploit the fact that videos are distributed in layers. Besides, we use a discarding policy based on those layers instead of discarding the whole video, as implemented on traditional web cache policy. We underline that the discarding of layers does not constrain a web cache to server requests to that video; that is, videos can be served, for instance, in a low definition instead of high definition. As a result, whenever a cache runs out of space and a new object needs to be cached, the number of cached videos increase since the discarding of a whole video will be postponed, at least in the first moment.

An important finding to be explained is how LRU policy and video layers discarding policy are related, as pointed in scenario number 2. In this scenario, we discard video layers that belong to the set of the least recently used videos. Nonetheless, this situation does not offer any improvement in both the number and quality of cached objects. This happens because the group of least used objects is very small (only popular video are cached), which heavily restrict the space where our discarding policy can act to improve the number of cached videos.

We noted as well that bringing video metadata (i.e., video popularity) into an object management policy associated with a global layers discard policy can increase substantially the quality and number of cached videos. Consequently, this improves the cache hit rate and saved traffic measured over the output link. Using these approaches, we got over 40% improvement which could mean to postpone updates over an output link.

7. Numerical Results

In this section, we have carried out some studies so as to evaluate the impact of the churn on object management policies implemented by peers. Accordingly, we have set up a peer-to-peer-assisted video distribution system, compare Section 4, and we have measured the decreasing ratio, that is, on how much the policy performance measured by the hit rate is impacted by the churn. The decreasing ratio is defined by the following equation:

$$\mathrm{HR}_{\mathrm{Decr}} = 1 - \frac{\mathrm{HR}_{\mathrm{With}}}{\mathrm{HR}_{\mathrm{Without}}}, \tag{6}$$

where $\mathrm{HR}_{\mathrm{With}}$ and $\mathrm{HR}_{\mathrm{Without}}$ are, respectively, the measured hit rates in a system with and without churn. In our experiments, we have employed both exponential and heavy tail churn models (cf. Section 4.2).

As we have pointed out, the accomplishment of an object management policy is affected by the performance metrics taken into account. For instance, the LRU policy can have high hit rate but performs poorly in term of byte hit rate. Therefore, we have studied also the decreasing ratio defined by the byte hit rate collected in a system with and without churn. From a qualitative stand point of view, the impacts measured by the decreasing ratio defined by both, hit rate and byte hit rate, are similar. Hence, we show as well the decreasing ratio defined by (6).

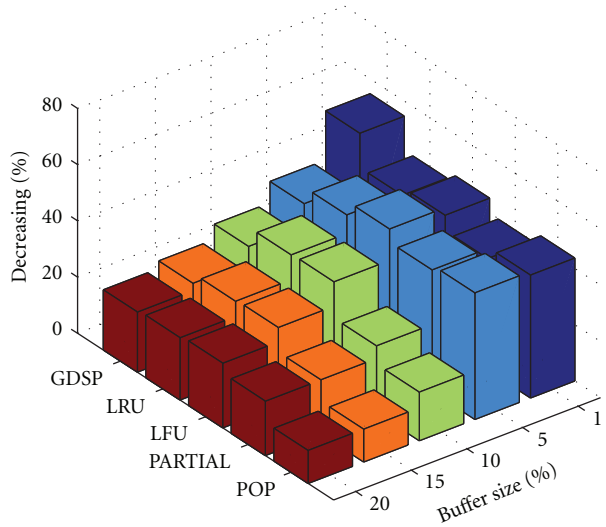
We have evaluated five policies: the context-aware and content-oriented policy (POP) [7], least recently used (LRU) policy, least frequently used (LFU) policy, popularity-aware greedy-dual size (GDSP) policy [1], and proportional partial caching (PARTIAL) policy [2]. The last four policies are representative implementations of their classes, that is, recency-based, frequency-based, and cost-based policies.

In our experiments, each peer can store up to 25 short videos of 4:50 minutes—the average video length identified in the studied video collection. The overall storage capacity, which is defined by the sum of the storage capacity of all peers, is equal to one percent of the video collection storage demand, see Section 3. We have defined different simulated scenarios by scaling up the system storage capacity until 20% of the video collection storage demand.

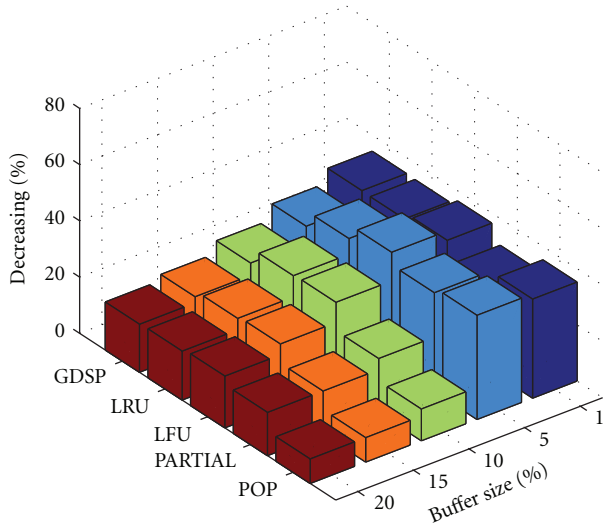
Figure 7 depicts the impact of the churn in term of decreasing on hit rate, when both exponential and heavy tail models drive the churn. Although their decreasing ratios, defined by (7), have different values, all policies' behavior is as follows: the gap among their performance shrinks as the system storage capacity is enlarged. However, this enlargement does not impact linearly the performance of policies. Specifically, for an enlargement of 20% in the system storage capacity, LFU policy has recorded a variation of 36% (29%) on its decreasing ratio for the exponential (heavy tail) churn model. LRU shows a variation of 34% (27%), while GDSP shows a variation of 33% (24%) and PARTIAL shows variation of 29% (23%).

The performance of POP policy represents an exception in the previous conclusion, specially for heavy tail churn model where the variation on its decreasing ratio is equal to 20%. That is, for the POP policy, the impact of churn is linearly reduced by the system enlargement. The volume of replicated data is the main reason for that performance (cf. Section 7.1).

The enlargement of the system storage capacity impacts policies performance, especially for exponential churn model where variations in the decreasing ratio are more pronounced as this enlargement happens. Figure 8 depicts the decreasing of system size, that is, the shrinking on the number of UP peers, when exponential and heavy tail churn models drive the joining and leaving events. When the churn is driven by the heavy tail model, the maximum decreasing on system size is 39%, whereas for exponential model, this value is 48%. As a general system design observation, we



(a) Exponential churn model



(b) Heavy tail churn model

FIGURE 7: Decreasing ratio of object management policies (see (6)).

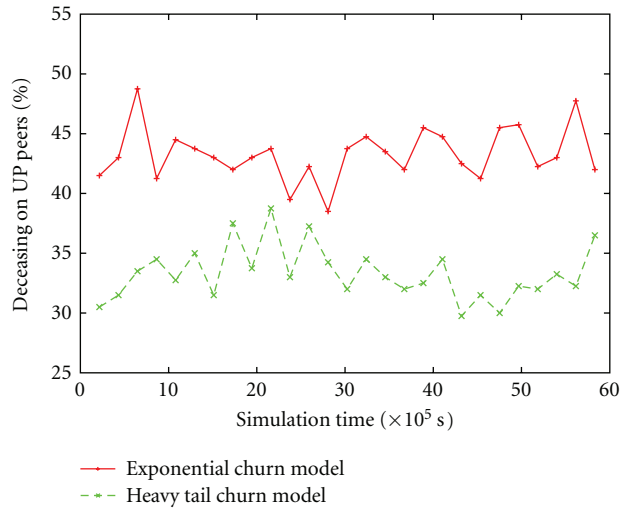


FIGURE 8: The decreasing on number of UP peers.

have noted that, since strong assumption on peers' storage capacity could jeopardize the system implementation, P2P-assisted systems have to deploy mechanisms to keep, over large time-scales, peers interested on content made available by communities.

7.1. Replicated Data. Each time a peer leaves the system, it will make chunks of popular content unavailable for other peers. Whenever that peer rejoins the system, its content could have been accessed by an other peer. This dynamic replicates data over peers that share interest and, consequently, decreases the nominal system storage capacity. Figure 9 shows the percentage of data that has been replicated into the system due to the churn.

For LRU, LFU, and GDSP policies, the maximum amount of replicated data demands 20% of storage capacity. However, for the two other policies, PARTIAL and POP, the amount of replicated data is around 25% and over 45%, respectively. The POP policy can handle twice as much replicated data than other policies. Since the POP policy keeps only objects with certain number of views (10,000 views in our experiments) and this requirement is too restrictive, the amount of replicated data grows rapidly due to the churn.

Resilience to failure is a key property of P2P systems. In this context, every time a peer A fails in delivering a service, which is under its responsibility, another peer will replace the peer A so as to deliver this service. This property has consequences in the proposed video distribution system; that is, the studied policies have to handle replicated data held independently by peers. From the preview results, at least 20% of stored data is replicated ones, this value being greater than 45% when the POP policy is used.

To measure the impact of replicating on policy performance, we have performed experiments by using the following nonreplicating procedure: peers must evict any replicated video every time they rejoin the system.

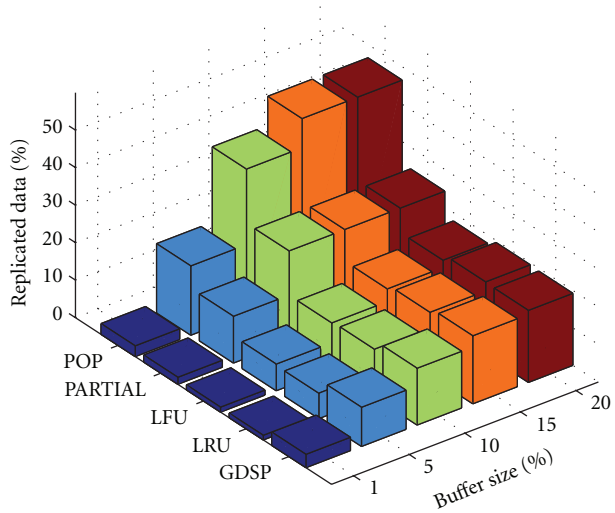
In this new scenario, we have computed the decreasing ratios as follows:

$$\mathrm{HR}_{\mathrm{DecrNorep}} = 1 - \frac{\mathrm{HR}_{\mathrm{NorepWith}}}{\mathrm{HR}_{\mathrm{Without}}}, \tag{7}$$

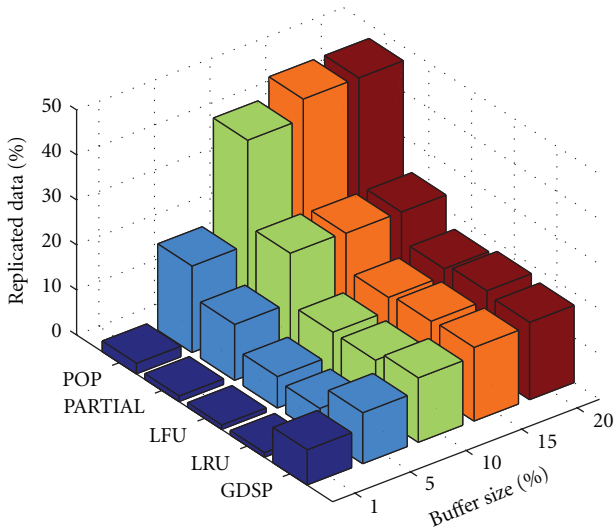
where $\mathrm{HR}_{\mathrm{NorepWith}}$ is the collected hit rate in a system with churn but without replicated data, and $\mathrm{HR}_{\mathrm{Without}}$ is the hit rate in a system without churn.

Figure 10 shows the decreasing ratio for all studies' policies. We have noted that in spite of the enlargement of the system storage capacities, the decreasing ratio is around 40% when the exponential churn model drives the joining and leaving dynamics. For peers under the heavy tail churn model, this decreasing ratio is around 20% (see Figure 10(b)).

Compared to results showed in Figure 7, the performance of policies under both churn models reduces by 50%.

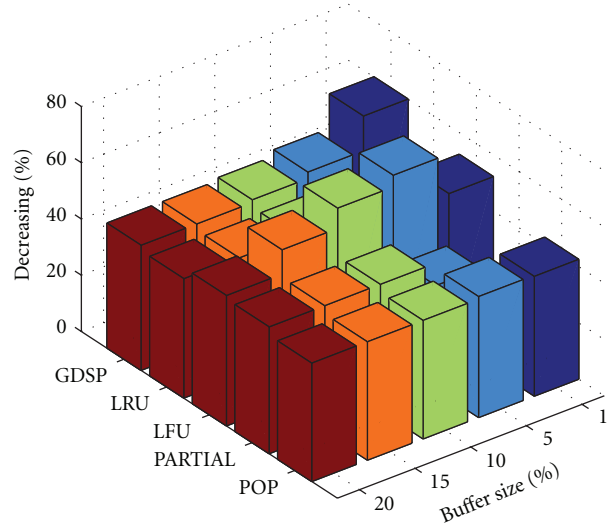


(a) Exponential churn model

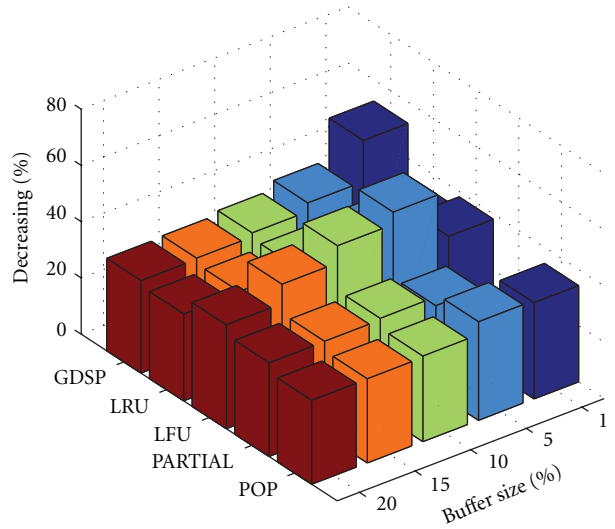


(b) Heavy tail churn model

Figure 9: The amount of replicated data handled by policies.



(a) Exponential churn model



(b) Heavy tail churn model

Figure 10: The decreasing on hit rate in a nonreplica system (see (7)).

This suggests that (i) an enlarged storage capacity has limited impact over policies performance and (ii) a controlled amount of replicated data tends to improve the performance of policies. In fact, it is under evaluation an algorithm to support replicating. Peers that just initiated the leaving procedure will use this algorithm to spread objects' metadata in order to support the decisions made by other peers during the evicting process.

8. Related Work

Several approaches (e.g., [8]) have been proposed in the literature so as to deal with video caching. In these approaches, whether a video or some part of it cannot be found in local proxies, it can be requested from the original central provider. In a P2P video system, however, there is no such central video server nor does any peer guarantee the availability of any data it caches.

Zink et al. [9] and Cha et al. [3] strongly suggest that metrics such as object popularity has to be considered in-cache object management policies. These studies show that popular and nonpopular videos access ratio is 2 in 3, despite popular video collection is being made of just 3.7% of the whole video collection. Although Zink et al. [9] show that local popularity has a weak correlation with the global popularity, still most of the accessed videos are the local popular ones.

Kulkarni and Devetsikiotis [10] propose the design of a distributed cache similar in principle to a content distribution network (CDN). The goal is to select some of the most valuable objects and bring them closer to the clients requesting them so that redundant load on the network is reduced. Social network analysis techniques were used to

identify the most valuable objects that should be cached. While Kulkarni's work focus is to determine the measurements that can be used to define the objects popularity, our goal is to verify the effectiveness of such measurements in the implementation of a cache approach based on P2P systems.

The behavior of P2P system under churn is one of the most fundamental issues of P2P research (e.g., [6, 11, 12]). Several approaches (e.g., [11, 13]) have dealt with churn by investigating its characteristics (e.g., median session length) in large-scale P2P systems. Gummadi et al. [11] measure session lengths by monitoring a router at the University of Washington. Sripanidkulchai et al. [13] study the live streaming workload of a large-content delivery system and present an analysis characterizing popularity, arrival process, and session length.

9. Conclusions

In this paper we studied object management policies in a peer-to-peer network context. While traditional object management policies were built around only three object proprieties, that is, aging, size, and frequency, we built our policy around user-generated content and metadata made available by content providers. We used video popularity and took advantage of video-encoding techniques to build our policy. We have carried out a set of simulation experiments so as to evaluate the performance of our approach. In the first part of our simulation experiments, we have observed that our approach outperforms the traditional one and consequently will reduce the capacity demand poses over the community output link by increasing the hit rate of community demands and optimizing the network resources available.

We have studied as well the impact of the churn on the object management policies. We evaluated the enlargement of peer's storage capacity as an action to keep policies performance since the system storage capacity will be affected by the churn. Though the gap among a policy's performance shrinks as we enlarged the storage capacity, it does not impact proportionally a policy performance.

Also, we have carried out studies to look into how replicated data impact the performance of object management policies. We found that the worst case scenario, in terms of content availability, is for a system without replicated data. Despite the enlargement of the storage capacity, in general, we have not seen improvements on policies performance for such systems. On the other hand, replicating due to the churn improved the performance of policies to a certain level. However, we have shown from the POP policies performance, whether the amount of duplicated data is under policy control, the content availability could be greatly improved.

As future works, we have looked into mechanisms to control the amount of replicated data through the system and investigated whether or not incentive-based peer participating mechanisms, developed for other content distribution systems, could be applied in such hybrid P2P-CDN system.

References

[1] S. Jin and A. Bestavros, "Popularity-aware greedy dual-size web proxy caching algorithms," in *Proceedings of the The 20th International Conference on Distributed Computing Systems (ICDCS '00)*, IEEE Computer Society, Washington, DC, USA, 2000.

[2] M. Hefeeda and O. Saleh, "Traffic modeling and proportional partial caching for peer-to-peer systems," in *Proceedings of the IEEE/ACM Transactions on Networking*, vol. 16, pp. 1447–1460, IEEE Press, Piscataway, NJ, USA, 2008.

[3] M. Cha, H. Kwak, P. Rodriguez, Y. Y. Ahnt, and S. Moon, "I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system," in *Proceedings of the 7th ACM SIGCOMM Internet Measurement Conference (IMC '07)*, pp. 1–14, October 2007.

[4] A. Crespo and H. Garcia-Molina, "Semantic overlay networks for P2P systems," in *Agents and Peer-to-Peer Computing*, vol. 3601 of *Lecture Notes in Computer Science*, pp. 1–13, Springer, Berlin, Germany, 2005.

[5] F. Wang, J. Liu, and Y. Xiong, "Stable peers: existence, importance, and application in peer-to-peer live video streaming," in *Proceedings of the 27th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '08)*, pp. 1364–1372, IEEE Computer Society, Phoenix, Ariz, USA, 2008.

[6] Z. Yao, D. Leonard, X. Wang, and D. Loguinov, "Modeling heterogeneous user churn and local resilience of unstructured P2P networks," in *Proceedings of the IEEE International Conference on Network Protocols, (ICNP '06)*, pp. 32–41, Washington, DC, USA, 2006.

[7] A. Bezerra, C. Melo, D. Vieira, Y. Ghamri-Doudane, and N. Fonseca, "A content-oriented web cache policy," in *Proceedings of the IEEE Latin-American Conference on Communications, (LATINCOM '09)*, pp. 1–6, September 2009.

[8] Z. Miao and A. Ortega, "Scalable proxy caching of video under storage constraints," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 7, pp. 1315–1327, 2002.

[9] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Watch global, cache local: youtube network traffic at a campus network—measurements and implications," in *Proceedings of the Multimedia Computing and Networking Conference, (MMCN '08)*, January 2008.

[10] V. Kulkarni and M. Devetsikiotis, "Communication timescales, structure and popularity: using social network metrics for youtube-like multimedia content distribution," in *Proceedings of the IEEE International Conference on Communications, (ICC '10)*, May 2010.

[11] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan, "Measurement, modeling, and analysis of a Peer-to-peer file-sharing workload," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles, (SOSP '03)*, pp. 314–329, ACM, New York, NY, USA, October 2003.

[12] D. Leonard, Z. Yao, V. Rai, and D. Loguinov, "On lifetime-based node failure and stochastic resilience of decentralized peer-to-peer networks," *IEEE/ACM Transactions on Networking*, vol. 15, no. 3, pp. 644–656, 2007.

[13] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An analysis of live streaming workloads on the internet," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference, (IMC '04)*, pp. 41–54, New York, NY, USA, October 2004.